UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/965,521	09/27/2001	Roch Georges Archambault	CA9-2000-0027	3302

7590 07/15/2004

Ronald L. Drumheller, Esq.
94 Teakettle Spout Road
Mahopac, NY 10541

EXAMINER	
RAMPURIA, SATISH	
ART UNIT	**PAPER NUMBER**
2124	

DATE MAILED: 07/15/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/965,521	ARCHAMBAULT, ROCH GEORGES	
	Examiner	Art Unit	
	Satish S. Rampuria	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *03* MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 September 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-14* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-14* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the application filed on 09/27/2001.

2. Claims 1-14 are pending.

Priority

3. Acknowledgment is made of applicant's claim for foreign priority under 35 U.S.C. 119(a)-(d). The certified copies have been received on 09/27/2001.

Claim objections

4. Claim 1, 3, 4, 10, 12, and 14 are objected to because of the following informalities:

Claims are objected to because of the following informalities:

Regarding claims 1 and 3 on page 13, lines 7 and 20 respectively, after the words "steps" and "steps of" respectively, missing ":".

Regarding claims 4 on page 14, line 7, after the word "steps of" missing ":".

Regarding claim 12, on page 16, line 18, after the word "comprising" missing ":".

Regarding claim 10 on page 15, lines 15 and 17 after the word "steps" missing ":" and after the word "definition by" missing ";" respectively.

Regarding claim 10 on page 15, lines 15 and 17 after the word "steps" missing ":" and after the word "definition by" missing ";" respectively.

Appropriate correction is required.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claim 1-7, 9, and 11-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,182,284 to Sreedhar et al., hereinafter called Sreedhar, in view of US Pub. No. 2002/0166115 to Sastry hereinafter called Sastry.

Per claims 1, 2, and 5:

Sreedhar disclose:

- A method for determining the correctness of a potential interprocedural dead store optimization for an optimizing compiler (col. 1, lines 13-15 "a method and system for translating optimized, intermediate-level, static-single-assignment-form computer code... instructions into optimized, intermediate-level computer code"), the optimizing compiler generating an intermediate representation of code to be compiled comprising a call graph (col. 4, lines 5-9 "control-flow-graph representation of SSA-form code, interference-graph representation of variable interferences in SSA-form code"), the method comprising a top-down traversal of the call graph, and comprising, for each procedure definition reached in the call graph traversal, the following steps
- a. defining a live on exit set of variables for the reached procedure definition (col. 9 lines 39-42 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic

block. A variable is "live"... a path to the exit of the program... value... used before it is redefined"), and

- b. defining a live on exit set of variables for each procedure call point within the reached procedure definition (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block"),
- d. using the live on exit set of variables for the reached procedure definition to determine the variables (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block") that are ineligible for interprocedural dead store elimination in the reached procedure definition (col. 9, lines 42-46 "A variable is "dead" at that point if there is no such path... a variable is live at the exit of a basic block if it is live at the entry to any of the basic block's successor basic blocks in the control flow graph").

Sreedhar does not explicitly disclose storing the live on exit set of variables for each procedure call point in an entry in a live on exit data structure.

However, Sastry discloses in an analogous computer system storing the live on exit set of variables for each procedure call point in an entry in a live on exit data structure (page 7, paragraph 106 "store instruction in the web that are live outside the interval, stores are inserted in the tail block of each exit edge of the interval").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of storing the live on exit set of variables as taught by Sastry into the method of optimization of a compiler as taught by Sreedhar. The

modification would be obvious because of one of ordinary skill in the art would be motivated to store the live on exit set of variables to ensure the value in the virtual register and in value in the memory are consistent before entering and after exiting during optimization as suggested by Sastry (page 1, paragraph 6 and 10).

Per claims 3 and 4:

The rejection of claim 2 is incorporated, and further, Sreedhar disclose:

- a. defining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition (col. 8, lines 66 to col. 9, lines 1-2 "The basic control flow graph, shown in FIG. 1… lists of variables, or registers, that are live at the beginning and the end of each basic block"), the basic block live set comprising the variables used in the block of computer code and the variables used in any procedure called within the block of computer code (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block"), and
- b. determining the live on exit set for each procedure call by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point (col. 9, lines 44-46 "a variable is live at the exit of a basic block if it is live at the entry to any of the basic block's successor basic blocks in the control flow graph") and by adjusting the union to include uses of variables in the code between the call point for the procedure and the end of the block containing the call point (col. 9, lines 18-22 "sets of variables called "USE(i)" for each basic block i are generated… set

"USE(i)" for basic block i contains variables, or registers, that are used before they are defined").

Per claim 6:

The rejection of claim 3 is incorporated, and further, Sreedhar disclose:

- which the variables used in a procedure called within a block of computer code are determined by accessing the mod/use set for the procedure associated with the procedure definition node in the call graph (col. 8, lines 49-55 "The nodes in the control flow graph of FIG. 1 correspond to basic blocks... edges linking the nodes... represent possible transfer of execution control by the last instruction of one basic block to the first instruction of another basic block")

Per claim 7:

The rejection of claim 1 is incorporated, and further, Sreedhar disclose:

- which the step of using the live on exit set of variables for the reached procedure definition to determine the variables (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block") that are ineligible for interprocedural dead store elimination in the reached procedure definition (col. 9, lines 42-46 "A variable is "dead" at that point if there is no such path... a variable is live at the exit of a basic block if it is live at the entry to any of the basic block's successor basic blocks in the control flow graph") comprises the step of generating pseudo uses of the members of the live on exit set of variables for the reached procedure definition in

the data flow graph for the reached procedure definition (col. 4, lines 17-20 "the redundant copy elimination problem with an example and provides a pseudo-code implementation of a method for identifying redundant copies")

Per claim 9:

The rejection of claim 2 is incorporated, and further, Sreedhar disclose:

- using the live on exit set of variables for the procedure definition to determine whether the procedure definition may be cloned by the optimizing compiler (col. 2, lines 63-67 to col. 3, lines 1-2 "Redundant copy instructions can be removed from the SSA-form intermediate-level code by considering the interference graph and by comparing the members of the phi congruence classes associated with the variables used in the copy instructions")

Claim 11 is the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claims 12 and 13 are the system claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 14 is the process claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

7. Claims 8 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sreedhar, Sastry in view of US Patent No. 5,175,856 to Van Dyke et al., hereinafter called Van Dyke.

Per claim 8:

The rejection of claim 1 is incorporated, and further, neither Sreedhar nor Sastry disclose the live on exit set data structure comprises bit vector entries and is indexed by call graph edges.

However, Van Dyke discloses in an analogous computer system the live on exit set data structure comprises bit vector entries and is indexed by call graph edges (col. 17, lines 24-27 "The symbol node 110 points to an array of bit vectors 150, each bit vector containing one entry for each block node 106 in the program. This array is indexed by the depth-first numbering of the block nodes 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of having an index in an array in a block nodes as taught by Van Dyke into the method of optimization of a compiler as taught in the combination system by Sreedhar and Sastry. The modification would be obvious because of one of ordinary skill in the art would be motivated to use bit vector entries and index by call graph to provide efficient method of optimization as suggested by Van Dyke (col. 4, lines 9-27).

Per claim 10:

Sreedhar disclose:

- A method for determining the correctness of a potential interprocedural dead store optimization for an optimizing compiler (col. 1, lines 13-15 "a method and system for

translating optimized, intermediate-level, static-single-assignment-form computer code… instructions into optimized, intermediate-level computer code"), the optimizing compiler generating an intermediate representation of code to be compiled comprising a call graph (col. 4, lines 5-9 "control-flow-graph representation of SSA-form code, interference-graph representation of variable interferences in SSA-form code"), the method comprising a top-down traversal of the call graph, and comprising, for each procedure definition reached in the call graph traversal, the following steps

- a. defining a live on exit set of variables for each procedure call point within the reached procedure definition by (col. 9 lines 39-42 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block. A variable is "live"… a path to the exit of the program… value… used before it is redefined")
- i. defining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition (col. 8, lines 66 to col. 9, lines 1-2 "The basic control flow graph, shown in FIG. 1… lists of variables, or registers, that are live at the beginning and the end of each basic block"), the basic block live set comprising the variables used in the block of computer code and the variables used in any procedure called within the block of computer code(col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block"), and
- ii. determining the live on exit set for each procedure call by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point (col. 9, lines 44-46 "a variable is live at the exit of a basic block if it is live at the entry to any of the basic block's successor basic blocks in the control

flow graph") and by adjusting the union to include uses of variables in the code between the call point for the procedure and the end of the block containing the call point (col. 9, lines 18-22 "sets of variables called "USE(i)" for each basic block i are generated... set "USE(i)" for basic block i contains variables, or registers, that are used before they are defined").

- c. defining a live on exit set of variables for the reached procedure definition (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block") by taking the union of all stored entries in the live on exit data structure corresponding to call points for the reached procedure (col. 9 lines 39-42 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block. A variable is "live"... a path to the exit of the program... value... used before it is redefined"),
- d. removing all entries in the live on exit data structure corresponding to call points for the reached procedure (col. 2, lines 65-66 "Redundant copy instructions can be removed from the SSA-form intermediate-level code"), and
- e. using the live on exit set of variables for the reached procedure definition to determine the variables (col. 9, lines 39-40 "LiveOut set is the set of variables, or registers, that are live at the exit of a basic block") that are ineligible for interprocedural dead store elimination in the reached procedure definition (col. 9, lines 42-46 "A variable is "dead" at that point if there is no such path... a variable is live at the exit of a basic block if it is live at the entry to any of the basic block's successor basic blocks in the control flow graph").

Sreedhar does not explicitly disclose storing the live on exit set of variables for each procedure call point in an entry in a live on exit data structure.

However, Sastry discloses in an analogous computer system storing the live on exit set of variables for each procedure call point in an entry in a live on exit data structure (page 7, paragraph 106 "store instruction in the web that are live outside the interval, stores are inserted in the tail block of each exit edge of the interval").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of storing the live on exit set of variables as taught by Sastry into the method of optimization of a compiler as taught by Sreedhar. The modification would be obvious because of one of ordinary skill in the art would be motivated to store the live on exit set of variables to ensure the value in the virtual register and in value in the memory are consistent before entering and after exiting during optimization as suggested by Sastry (page 1, paragraph 6 and 10).

Neither Sreedhar nor Sastry disclose a bit vector indexed by a call graph edge.

However, Van Dyke discloses in an analogous computer system a bit vector entries and is indexed by call graph edges (col. 17, lines 24-27 "The symbol node 110 points to an array of bit vectors 150, each bit vector containing one entry for each block node 106 in the program. This array is indexed by the depth-first numbering of the block nodes 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of having an index in an array in a block nodes as taught by Van Dyke into the method of optimization of a compiler as taught in the

combination system by Sreedhar and Sastry. The modification would be obvious because of one of ordinary skill in the art would be motivated to use bit vector entries and index by call graph to provide efficient method of optimization as suggested by Van Dyke (col. 4, lines 9-27).

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

The following patent is cited to further show the state of the art with respect to optimizing compiler.

US Patent No. 5,878,261 to Holler et al.

US Patent No. 5,812,855 to Hiranandani et al.

US Patent No. 5,555,417 to Odnert et al.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Satish S. Rampuria whose telephone number is 703-305-8891. The examiner can normally be reached on 8:30 am to 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703) 305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner
Art Unit 2124
07/12/2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100